11-2022

Which neural network makes more explainable decisions? An approach towards measuring explainability

Mengdi ZHANG
Singapore Management University, mdzhang.2019@phdcs.smu.edu.sg

Jun SUN
Singapore Management University, junsun@smu.edu.sg

Jingyi WANG
Zhejiang University

Citation

ZHANG, Mengdi; SUN, Jun; and WANG, Jingyi. Which neural network makes more explainable decisions? An approach towards measuring explainability. (2022). *Automated Software Engineering*. 29, (2), 1-26.
Available at: https://ink.library.smu.edu.sg/sis_research/7160

Published in Automated Software Engineering. November 2022, Vol. 29 Issue 2. pp. 1-26.
https://doi.org/10.1007/s10515-022-00338-w

Which neural network makes more explainable decisions? An approach towards measuring explainability

Mengdi Zhang[1] · Jun Sun[1] · Jingyi Wang[2]

Abstract
Neural networks are getting increasingly popular thanks to their exceptional performance in solving many real-world problems. At the same time, they are shown to be vulnerable to attacks, difficult to debug and subject to fairness issues. To improve people's trust in the technology, it is often necessary to provide some human-understandable explanation of neural networks' decisions, e.g., why is that my loan application is rejected whereas hers is approved? That is, the stakeholder would be interested to minimize the chances of not being able to explain the decision consistently and would like to know how often and how easy it is to explain the decisions of a neural network before it is deployed. In this work, we provide two measurements on the decision explainability of neural networks. Afterwards, we develop algorithms for evaluating the measurements of user-provided neural networks automatically. We evaluate our approach on multiple neural network models trained on benchmark datasets. The results show that existing neural networks' decisions often have low explainability according to our measurements. This is in line with the observation that adversarial samples can be easily generated through adversarial perturbation, which are often hard to explain. Our further experiments show that the decisions of the models trained with robust training are not necessarily easier to explain, whereas decisions of the models retrained with samples generated by our algorithms are easier to explain.

Keywords Neural network testing · Model interpretability · Deep learning models

✉ Jun Sun
junsun@smu.edu.sg

Mengdi Zhang
mdzhang.2019@phdcs.smu.edu.sg

Jingyi Wang
wangjyee@zju.edu.cn

[1] Singapore Management University, Singapore, Singapore

[2] Zhejiang University, Hangzhou, China

1 Introduction

Neural networks are getting increasingly popular thanks to their exceptional performance in solving many real-world problems, such as face recognition (Tran et al. 2017), language translation (Lewis et al. 2019) and self-driving cars (Bojarski et al. 2016). At the same time, multiple issues have been identified as well. For instance, it has been shown that neural networks are subject to adversarial sampling attacks, i.e., a correctly classified sample would be easily misclassified once some perturbation is applied (Goodfellow et al. 2014). For another instance, a neural network model may be embedded with a backdoor to predict differently in the presence of certain trigger (Liao et al. 2018) or may be discriminative against certain groups (Mohammed et al. 2020).

One way to reduce people's distrust in neural networks is to make sure that there is a human-understandable explanation of the neural network's decisions. Consider a scenario where a neural network is used to decide whether a bank loan application is approved or rejected. It would be an issue if we cannot explain why one application is rejected whereas a similar one is approved. Worse yet, it becomes a fairness issue if the only explanation possible is based on certain sensitive features (such as race and gender). A stakeholder thus would be interested to minimize the chances of not being able to explain the decision consistently. How often and how easy it is to explain the decisions of a neural network would be significant evaluation criteria. In other words, there is a need to test neural networks in terms of the explainability of their decisions. To the best of our knowledge, this is a testing problem which is yet to be addressed.

Since we aim to explain decisions from users' perspectives, the need is closely related yet different from existing studies on neural network interpretability. It is based on recent studies on what are considered human-understandable. These studies are typically conducted through human studies (Lombrozo 2007; Oaksford and Chater 2001; Lage et al. 2019). The conclusion is that comparing different models for explanation, human understanding goes only as far as simple models such as decision trees and linear functions. The consequence is that when we explain a neural network's decisions, we are limited to simple linear functions (e.g., "your loan is rejected before your annual income is below 50K whereas hers is more than 50K") or simple decision trees (e.g., "your loan is rejected because your annual income is below 50K and you have an existing loan, whereas although her annual income is similar, she does not have an existing loan"). Note that this need (of providing a human-understandable explanation of neural network decisions) is different from the need addressed by works which aim to provide interpretation on the inner working of neural networks (Kim et al. 2014; Lakkaraju et al. 2016), such as the studies in Simonyan et al. (2013), Yosinski et al. (2015), and Guo et al. (2018) which provide hints on how a neural network makes certain decision through highlighting certain pixels of the input picture or certain words of an input text.

In this work, we develop an approach and a software toolkit to address the problem. We first propose multiple measurements on how easy it is to explain a

user-provided neural network's decisions based on the results of the above-mentioned user studies (Lombrozo 2007, 2006; Oaksford and Chater 2001; Lage et al. 2019). Intuitively, we define the measurements based on how often the prediction results can be explained consistently using a human-understandable model and how simple the model is. These measurements allow us to compare different neural networks in terms of how explainable their decisions are. Afterwards, we develop multiple algorithms that allow us to automatically evaluate and compare neural networks in terms of the measurements. Given a neural network, our algorithms systematically explore the input space and measure the percentage of inputs on which the prediction results can be consistently explained with a certain simple model. We remark that our approach can be regarded as a generalization of the recent work on evaluating fairness (Agarwal et al. 2018; Zhang et al. 2020) (where fairness is defined in terms of individual discrimination). That is, a fairness issue is a special case of a decision explainability problem, i.e., the only possible explanation is based on the sensitive features. In our testing,

Our approach has been implemented as a self-contained toolkit (open-source at Zhang (2021)). We conduct multiple experiments on neural network models trained based on benchmark datasets to evaluate the relevance of our approach and the effectiveness of our algorithms. Our experiment results show that, unsurprisingly, the neural network models almost always have a low decision explainability. One of the reasons seems to be the existence of adversarial samples. We further perform experiments to check whether models obtained through robust training have improved decision explainability. The results suggest that it is not always the case. Lastly, we check whether we can improve the decision explainability of a model by retraining it using samples generated by our algorithms without reducing accuracy significantly, and the results are affirmative.

The rest of the paper is organized as follows. In Sect. 2, we define our problem and provide multiple measurements of the decision explainability of neural networks. In Sect. 3, we present our algorithms for evaluating neural networks based on our measurements. In Sect. 4, we evaluate our approach through multiple experiments. Lastly, we review related work in Sect. 5 and conclude in Sect. 6.

2 Problem definition

In this work, our goal is to develop systematic methods to measure how easy it is to explain the decision of neural networks in a human-understandable way. To define the problem properly, we must answer questions such as: what are considered human-understandable and what quantitative measurements do we use? In the following, we first review existing related literature and then define the problem.

2.1 What are human-understandable?

There are multiple human studies on what are considered human-understandable. For example, it is observed that human beings prefer explanations that are both

simple and highly probable (Lombrozo 2007) and often develop simplified understanding of complex systems by ignoring low-probabilistic cases (Oaksford and Chater 2001). One study compares the effectiveness of different forms of explanations by measuring human response time (Lage et al. 2019). The result shows that simulation was the fastest, followed by verification and then counterfactual reasoning. The counterfactual reasoning task also has the lowest accuracy.

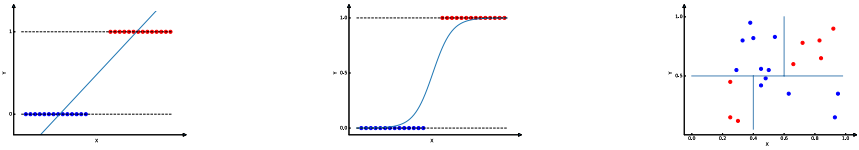
Recent studies (Carvalho et al. 2019) found that human-understandable models include linear regression, logistic regression, and decision trees. The linearity of the learned relationship makes human-understanding easy, especially for the model with monotonous constraints (Carvalho et al. 2019).

Logistic regression is an extension of linear regression, typically for classification problems. Instead of fitting a straight line or a hyperplane, it outputs the prediction probabilities between 0 and 1 for different classes. While linear regression and logical regression are limited to linear relationships, decision trees can be used to represent non-linear relationships to some extent (i.e., some Boolean combinations of them). A decision tree model splits the data multiple times according to certain atomic propositions on the input features.

Figure 1 shows one sample model for each kind. The task is to classify the blue and red dots. Given the task, a linear regression, shown in Fig. 1a, is a best-fit straight line which aims to separate the dots based on predicted value; a logistic regression, shown in Fig. 1b, takes the weighted sum of the inputs and applies an activation function such as Sigmoid to generate classifications; and a decision tree, shown in Fig. 1c, separates the input space into multiple regions and each of the regions is assigned with one predicted classification label.

Intuitively, our idea of defining and measuring how easy it is to explain the decisions of a neural network model is based on measuring how often the prediction results can be "explained" consistently using a human-understandable model. Existing user studies suggest that humans are capable of understanding simple models such as the kind of decision trees that we focus on Breiman (1984). Thus, by measuring how often the neural network's decisions can be explained by such a simple model, we get a measure on how often people can understand the neural network's decisions.

In this work, we focus on decision trees over linear regressions for the following reasons. First, as samples end up in distinct groups according to a decision tree model, it is arguably easier to understand than points on lines or multi-dimensional hyperplanes as in linear regression models. Secondly, the tree structures of decision



(a) Linear regression (b) Logistic regression (c) Decision tree

Fig. 1 Human-understandable models

trees have a natural visualization and humans can reason about its decision-making process following its hierarchical structure. Thirdly, decision trees are reasonably expressive (e.g., they are more expressive than linear regressions and they can capture certain non-linear relationships) and often achieve more accurate prediction results on real-world tasks (Zhang et al. 2019; Ke et al. 2017).

2.2 Learning decision trees

Note that our approach is not approximating a given neural network using decision tree, but quantitatively evaluating neural network explainability based on decision tree (or any other simple approximation model) which works as a proxy. So our approach relies on methods for generating decision trees. In the literature, there are well-studied algorithms for generating decision trees (Steinberg and Colla 2009; Chen et al. 2016; Ke et al. 2017). In this work, we adopt the existing method called the classification and regression trees (CART (Breiman 1984))to construct decision trees. CART relies on the learning samples which are a set of historical data with assigned classes for all observations to learn decision trees. It raises a sequence of yes/no questions, each of which splits the learning samples into two partitions. In other words, given a set of labeled feature vectors, the algorithm identifies one atomic proposition each time for splitting the domain of each feature. The atomic proposition is identified greedily by minimizing the Gini score, which is defined in David (1980). In the following, we provide a formal definition of decision trees and refer the readers to the existing literature on how to learn decision trees.

Definition 1 (Decision trees) Given a set of atomic propositions $\psi_1, \psi_2, \cdots, \psi_m$, a (binary) decision tree is a binary tree where each node has two outgoing edges, one labeled with a proposition ψ_i and the other labeled with $\neg\psi_i$. □

A sample decision tree is shown in Fig. 2. If a decision tree has height n, there are a maximum of 2^n leaf nodes. Each leaf node is associated with a prediction. Given a sample, the prediction by the tree can be identified by navigating from the root of the tree and following the edges according to the truth value of the proposition associated with the node, until a leaf node is reached.

The above definition is parameterized with a set of propositions. The choice of propositions would obviously be related to whether the decision tree is considered human-understandable. In this work, we restrict propositions to the form

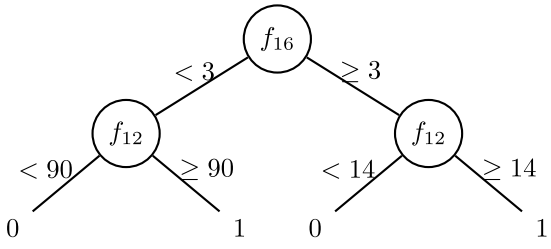


Fig. 2 A sample decision tree on the Bank dataset

$f \geq d$ where f is a feature and d is a constant, e.g., *income* $\geq 50K$. We choose Boolean proposition for linear relationship instead of continuous in the range, e.g., *income* $\in [50K, 60K]$, because the former proposition always splits the space into two partitions which is easier for human reasoning and two such propositions work the same as continuous in the range. We leave it to future work to consider propositions relating to multiple features.

Algorithm 1 $buildTree\,(LD, KF, K)$ where LD is a set of labeled data and KF is a set of features

1: let $node$ be a new node
2: **if** K = 0 or all samples in LD have the same label **then**
3: **return** $node$
4: **else**
5: identify a proposition ψ of the form $f \geq d$ such that $gini_{\psi}$ is minimum for all $f \in KF$ and d
6: let $left$ be the samples in LD that violates ψ
7: let $right$ be the samples in LD that satisfies ψ
8: let left child of $node$ be $buildTree\,(left, KF, K-1)$ and right child be $buildTree\,(right, KF, K-1)$
9: **end if**

The details of the decision tree building algorithm are shown in Algorithm 1. We limit the height of the tree to be maximum *K*, which is an input parameter, along with a set of labeled samples *LD* and features *KF*. Note that, instead of letting CART algorithm do the feature selection, we use *KF* to define features used to approximating the neural network. In this setting, we support evaluating how easy it is to explain neural network's decisions using any user-provided feature set and also ignore the possible effect of sensitive features, e.g., gender or race. In the base case, if $K = 0$, we simply return a new node, which will be a leaf of the decision tree. Otherwise, we identify a proposition ψ at line 5 (by trying all feature f and constant d in f's domain). Afterwards, the set of labeled samples is split into two, i.e., *right* contains those which satisfy ψ, and *left* contains the rest. A perfect split is one that all samples in *left* have the same label and all samples in *right* have the same label. In such a case, the Gini score is 0. Since we exhaustively search for the ψ which has the minimum score (assuming that there are finitely many d values) for each level of the decision tree, the generated decision tree is guaranteed to have better accuracy on *LD* than any other decision trees. This is stated in the following theorem.

Theorem 1 (Breiman 1984) *If Algorithm* 1 *returns a decision tree D with accuracy* ϕ, *that does not exist a decision tree D with the same height and accuracy* ϕ' *such that* $\phi' > \phi$.

2.3 Defining measurements

In the following, we define multiple measurements that can be used to quantify how easy it is to explain the decisions of a neural network and also how easy it is for human to understand neural network's decisions. Note that we evaluate on the predictions of the neural network instead of the ground truth labels.

Definition 2 (K-explainability) A model N's decisions are K-explainable with respect to a set of samples T iff there exists a decision tree D such that $height(D) \leq K$ and $D(i) = N(i)$ for all $i \in T$, where $N(i)$ is the prediction on i by N and $D(i)$ is the prediction on i by D. □

Intuitively, a model N's decisions are K-explainable if and only if its decisions can be consistently explained using a decision tree with a height no more than K. In the above definition, the decision explainability is parameterized with a limit on the height of the decision tree K. It is known (Ke et al. 2017) that the higher the decision tree is, the less understandable it is. Thus, we must take the height of the decision tree into account. Furthermore, the above is defined with respect to a set of samples T, i.e., the set of samples on which we evaluate the model's decision explainability. In this work, we focus on two sets. One is the training set, i.e., the decision explainability of a model is tested and measured against those samples in the training set. The other is the entire input space, i.e., the decision explainability of a model is tested against any sample, including those that are yet to be seen. The latter is clearly more demanding than the former. Given the above definition, our problem is to test whether a model N's decisions are K-explainable. We show an algorithm to solve the problem in Sect. 3.

In the following, we define K, ϕ-explainability which is a slightly relaxed notion of decision explainability. That is, instead of demanding that a neural network is completely consistent with a decision tree with a limited height (which is often unlikely in practice as we show in Sect. 4), we require that the explaining model must reach a certain level of consistency with the neural network model in terms of the decisions.

Definition 3 (K, ϕ-explainability) Let ϕ be a percentage. A model N's decisions are K, ϕ-explainable with respect to a set of samples T if and only if there exists a decision tree D such that $height(D) \leq K$ and the fidelity of D with respect to N is no less than ϕ where fidelity is defined as follows.

$$Fidelity = \frac{\sum_{i \in T} Sign(N(i) = D(i))}{\#T} \tag{1}$$

where $Sign(b)$ is a function which returns 1 if b is true.

Intuitively, a model N's decisions are K, ϕ-explainable if and only if its decisions can be explained consistently using a decision tree with a height no more than K in most of the cases (defined by ϕ). The value of ϕ provides a quantification of the

explainability. Note that we similarly distinguish two cases based on what T refers to. One is the training set and the other is the entire input space. Note that in the latter case, since there may be infinitely many samples, measuring the fidelity is non-trivial. To evaluate whether a model's decisions are K, ϕ-explainable, we would like to have an algorithm which can systematically measure the fidelity of the best decision tree with a height no more than K. We present such an algorithm in Sect. 3.

With the above-defined measurements, we can then compare the decision explainability of two neural network models N_1 and N_2. We say that N_1 is more explainable than N_2 if N_1's decisions are K_1-explainable, N_2's decisions are only K_2-explainable and $K_1 < K_2$; or N_1's decisions are K, ϕ_1-explainable, N_2's decisions are only K, ϕ_2 -explainable and $\phi_1 > \phi_2$.

3 Testing decision explainability

In this section, we answer the following question: how do we measure the decision explainability of a user-provided neural network model? We present four algorithms according to the two measurements defined in Sect. 2.

3.1 *K*-Explainability testing against training set

In the following, we present algorithms for testing whether a neural network's decisions are *K*-explainable. We distinguish two cases. One is that the decision explainability is defined with respect to the training set. The other is that it is defined with respect to the entire input space. The former is solved using Algorithm 2.

Algorithm 2 $isExplainableTrainSet\,(K, T, N)$

1: label T using N to obtain a labeled dataset LD
2: sets = all combinations of K features
3: **for** each set of features KF in sets **do**
4: $D = buildTree\,(LD, KF, K)$
5: **if** $\forall\, x$ in LD, $D\,(x) = N\,(x)$ **then**
6: **return** $true, D$
7: **else**
8: **return** $false, None$
9: **end if**
10: **end for**

Intuitively, the goal of Algorithm 2 is to test whether there exists a decision tree of height K which perfectly explains every prediction of the neural network. It takes the neural network model N and the training set T as well as K as inputs. At line 1, we label each feature vector in T with the corresponding prediction of the neural network N. Note that our goal is to explain N's decisions and thus the ground truth labels of T are irrelevant here.

At line 2, we identify all combinations of K-features. Suppose each input feature vector has a total of M features. Then there are C_M^K different combinations of K features. The loop from line 3 to 10 tries every combination one by one. Note that this loop can be easily parallelized. For each combination, Algorithm 1 is applied to generate the 'best' decision tree using the given set of features. If perfect accuracy is achieved, i.e., every feature vector is classified with a label that is consistent with that of N, a decision tree satisfying Definition 2 with respect to the training set is identified and thus the neural network model's decisions are K-explainable. Furthermore, the generated decision tree is reported as evidence of the model's decision explainability. The complexity of the testing is $\Theta(C_M^K)$, where M is the dimension of feature vectors and K is the depth of decision trees. We remark that the complexity can be high when M is large and we experiment heuristics in Sect. 4 which may significantly reduce the complexity.

The following theorem states the soundness and completeness of the algorithm, whose correctness can be easily established based on Theorem 1.

Theorem 2 *A neural network N's decisions are K- explainable with respect to the training set if and only if Algorithm* 2 *returns true*. □

Example 3.1 We use the **Bank** (Moro et al. 2014) dataset to illustrate Algorithm 2. Each sample in the dataset has 16 features. Assume K is 2 and there are C_{16}^2 different combinations of 2 features. Next, each combination is tested. Let us take the combination of f_{12} (with domain $0-99$) and f_{16} (with domain $0-3$) as an example. The decision tree generated at line 4 is shown in Fig. 2.

Based on the decision tree, we have 3 propositions. ψ_1 means $f_{16} < 3$, ψ_2 means $f_{12} < 90$ and ψ_3 means $f_{12} \geq 14$.

We then travel through all samples in the training set to see whether the decision tree is completely consistent with the neural network. If we find a sample x such that the prediction of the neural network and decision tree is different, the neural network is considered not explainable according to this decision tree. That is, the decisions of the neural network cannot be explained entirely based on the values of the feature f_{12} and f_{16}. The following is such an example, whose prediction according to the neural network is 1 whereas its prediction according to the decision tree is 0.

$x = [3, 7, 0, 2, 0, 1, 0, 0, 0, 8, 4, 81, 1, 1, 0, 0]$

3.2 *K*-Explainability testing against all inputs

It may be insufficient to explain a model's decisions over the training set only, as a future sample might not fit in the explanation. In the following, we present an approach to test the explainability of a model N's decisions against all possible inputs including the unseen ones. In this setting, applying Algorithm 2, which requires us to exhaustively enumerate all possible inputs, is infeasible due to the enormous input space.

We use a best-effort approach to solve the problem. That is, we first sample (uniformly in the entire input sample space) a certain number of feature vectors and their corresponding labels (i.e., predictions made by N). Note that our algorithms can be readily adopted to a prior distribution (such as a normal distribution) is provided. We remark that it is hard to define the actual feature distributions and we view the problem of identifying the actual feature distribution as a problem that is orthogonal to our problem. After sampling, we learn a decision tree that is best consistent with the samples and then test whether the decision tree is consistent throughout the sample space. The details of the algorithm are shown in Algorithm 3.

It takes N as well as K as inputs. At line 1, we first generate a set of feature vectors as X from the entire input space randomly. Note that our approach can be easily modified to sample according to the data distribution of the training set if it is provided (or extracted using existing approaches (Shore 1998; Zhang et al. 2009)). Then at line 2, we label all feature vectors in X with the prediction of N to obtain a set of labeled samples LD. Afterwards, the same as in Algorithm 2, we identify all combinations of K-features at line 3. Then we test every combination in the loop from line 4 to 9. For each combination, Algorithm 1 is applied to build the 'best' decision tree. If the decision tree fails to achieve 100% accuracy on LD, we proceed to try the next combination. Otherwise, we further test whether the decision tree is applicable to the entire input space using Algorithm 4.

Algorithm 4 is inspired by existing approaches on adversarial perturbation (Goodfellow et al. 2014; Kurakin et al. 2016) with a different objective function. That is, the goal is to apply perturbation to samples in LD so as to find a 'counterexample', i.e., one such that the prediction of N and that of the decision tree are different. First, we cluster those samples in LD using a standard clustering algorithm k-means at line 1, where $\#c$ is the size of clusters (Lloyd 1982). Note that the reason for clustering is that we can diversify the search for counterexamples. Afterwards, we obtain seed samples from each cluster in a round-robin fashion at line 2. For each seed sample, we apply a gradient-based algorithm to search for counterexamples iteratively (see loop at line 4 to 11). The gradient is commonly used to generate adversarial samples (Pei et al. 2017; Moosavi-Dezfooli et al. 2016). The intuition is to perturb the original input according to the direction of the gradient so that the prediction of the neural network has the maximum change. During each iteration of the loop, we calculate the gradient of the loss function on the input x as J(x) and calculate sign of the gradient as $sign$(J(x)). Here, the *sign* function is used to extract the sign of the real gradient. Then we perturb the seed sample x with an amount $step\_size * grad$, where $step\_size$ is used to determine the perturbation degree in each time and $grad$ is the gradient.

Note that Algorithm 4 is a best-effort approach as we limit the number of iterations at line 4. If we find a counterexample, the algorithm returns *false*; otherwise, it returns *true*. Only if Algorithm 4 fails to find a counterexample, Algorithm 3 reports *true* with D at line 7.

Algorithm 3 $isExplainableAllInput\,(K, N)$

generate sample set X from input space randomly
label X using N to obtain a labeled dataset LD
$sets$ = all combinations of K features
for set KF in $sets$ **do**
  $D = buildTree\,(KF, LD, K)$
  if $\forall\, x$ in LD, $D\,(x) = N\,(x)$ and $attackFail\,(LD, D)$ **then**
    return $true, D$
  end if
end for
return $false, None$

Algorithm 4 $attackFail\,(LD, D)$ where $\#c$ is constant

clusters = KMeans(LD, $\#c$)
for i from 0 to p_num **do**
  Get seed x from clusters in a round-robin fashion
  for iter from 0 to max_iter **do**
    $grad = sign(\mathrm{J}\,(\mathrm{x}))$
    $x' = x + step\_size * grad$
    if x' is valid and $N\,(x') \neq D\,(x)$ **then**
      return $false$
    end if
    $x = x'$
  end for
end for
return $true$

Example 3.2 We illustrate how Algorithm 3 works using the **Bank** dataset example based on the feature combination (f_{12}, f_{16}). We first generate 5000 samples randomly and generate a decision tree *D* which is consistent with the neural network predictions. We then apply Algorithm 4 to test whether *D* is consistent throughout the input space. Assume that we take the following *x* as a seed.

x : [5, 10, 1, 1, 0, 19, 1, 0, 0, 5, 4, 81, 1, 1, 0, 0]

Intuitively, the goal is to perturb x such that its label changes and hopefully becomes different from that of *D*. The perturbation is guided by the gradient (so that the perturbation would cause a maximum change in the prediction). That is, we calculate the sign of *x*'s gradient as follows.

grad : [1, 1, 1, −1, 1, 1, −1, 1, 1, −1, 1, 0, −1, −1, −1, 0]

Next, we perturb x by updating each feature f_i to be $f_i + step\_size * grad[i]$ where *step_size* is 1 in this example. The result is the following sample.

x' : $[6, 11, 2, 0, 1, 20, 0, 1, 1, 4, 5, 81, 1, 0, 0, 0]$

Note that a clipping function is applied to make sure the perturbed feature value remains in its domain. The predictions of this sample by N and D are 0 and 1 respectively. As a result, D fails to be taken as an explanation of the neural network N.

3.3 K, ϕ-Explainability testing

As discussed above, due to the gap between the expressiveness of decision trees and neural networks, there is likely no decision tree (with a height limit such as K) that can be used to perfectly explain the decisions of the neural network. It is thus more useful to quantify the explainability by testing K, ϕ-explainability of the given neural network in practice. In the following, we similarly distinguish two cases, i.e., the training set and the entire input space. The former is addressed using Algorithm 5. The latter is addressed using Algorithm 6.

Algorithm 5 $explainabilityTrainSet\,(K, T, N, \phi)$

1: label T using N to obtain a labeled dataset LD
2: sets = all combinations of K features
3: **for** each set of features KF in sets **do**
4: $D = buildTree\,(LD, KF, K)$
5: **if** accuracy of D is no less than ϕ **then**
6: **return** $true, D$
7: **end if**
8: **end for**
9: **return** $false, None$

Algorithm 5 is similar to Algorithm 2. The only difference is that once Algorithm 1 is applied to generate the 'best' decision tree, the fidelity of the decision tree is compared with the given threshold ϕ. If a decision tree with a fidelity no less than ϕ is identified, the algorithm returns *true* together with the decision tree. The following theorem states the soundness and completeness of the algorithm, whose correctness can be easily established based on Theorem 1.

Theorem 3 *A neural network N's decisions are K, ϕ-explainable with respect to the training set if and only if Algorithm* 5 *returns true*. □

Example 3.3 Assume that we are testing whether a neural network trained on the **Bank** dataset is 2, 90%-explainable. Further, assume that we are testing the combination of f_{12} and f_{16}. Applying Algorithm 1, we obtain the decision tree shown in Fig. 2, which has an accuracy of 94.92% (calculated based on *LD*). As a result,

Algorithm 5 returns *true*. That is, we can explain 90% of the decisions of the neural network (over the training set) using a simple explanation based on the decision tree.

Testing whether a model's decision is K, ϕ-explainable with respect to all inputs is challenging as it is infeasible to exhaustively enumerate all inputs and check whether there exists a decision tree with prediction accuracy more than ϕ. We thus take a best-effort approach similar to that of Algorithm 3. That is, we uniformly sample a certain number of samples and build a candidate decision tree. Afterwards, we evaluate whether this candidate has a fidelity of ϕ throughout the input space. We formulate the latter problem as a hypothesis evaluation problem and solve it systematically using hypothesis testing (so that we have a certain level of statistical confidence in the evaluation result (Wald 1945)). That is, given the candidate decision tree D and the threshold ϕ, we formulate two hypotheses. One is that the fidelity of D is no less than ϕ and the other is that it is less than ϕ. We then keep sampling randomly (in an IID manner (Clauset 2011)) from the entire input space until one of the hypotheses reaches a certain level of statistical confidence.

Algorithm 6 $explainabilityAllInput\,(K, N, \phi)$

generate sample set X from input space randomly
label X using N to obtain a labeled dataset LD
$sets$ = all combinations of K features
for set KF in $sets$ **do**
  $D = buildTree\,(KF, LD, K)$
  if $SPRT\,(N, D, \phi)$ **then**
    return $true, D$
  end if
end for
return $false, None$

Algorithm 7 $SPRT\,(N, D, \phi)$

1: $p_0 = \phi - \sigma$
2: $p_1 = max(0.99, \phi + \sigma)$
3: stop = False
4: **while** not stop **do**
5: generate sample x from input space randomly
6: n = size of totally detected samples X
7: s = size of $x \in X | N\,(x) = D\,(x)$
8: $sprt\_ratio$ = calculate_sprt(s, n, p_0, p_1)
9: **if** $sprt\_ratio \geq \frac{1-\beta}{\alpha}$ **then**
10: **return** accept
11: **end if**
12: **if** $sprt\_ratio \leq \frac{\beta}{1-\alpha}$ **then**
13: **return** deny
14: **end if**
15: **end while**

The details of the algorithm used to evaluate whether a candidate decision tree achieves a fidelity of ϕ against all inputs are shown in Algorithm 6. It is designed based on the sequential probability ratio test (SPRT) algorithm proposed in Wald and Wolfowitz (1948). SPRT decides whether to accept a hypothesis after evaluating every new sample. The SPRT algorithm is parameterized by α, β and indifference region σ, which intuitively define the error bounds. According to ϕ and σ, we calculate p_0 and p_1 in line 1 and 2. During the loop from line 4 to 15, we keep generating new samples (i.e., by randomly generating values for each feature from its domain). Then we detect whether labels predicted by *N* and *D* are the same and update *n* and *s* respectively (see in line 6 and 7). Then we calculate the SPRT probability ratio at line 8 using the following formula.

$$sprt\_ratio = \frac{p_1^s \left(1 - p_1\right)^{n-s}}{p_0^s \left(1 - p_0\right)^{n-s}} \tag{2}$$

From line 9 to 14, we compare the SPRT ratio with the acceptance/rejection bounds. The test accepts the hypothesis if the SPRT ratio is larger than or equal to the acceptance bound or denies it if the SPRT ratio is less than or equal to the rejection bound. The algorithm stops whenever a hypothesis is accepted at line 10 or denied at line 13. Note that this algorithm is guaranteed to terminate and the probability of accepting the wrong hypothesis is bounded (Wald and Wolfowitz 1948).

Example 3.4 Let us use the **Bank** dataset and the feature combination of f_{12} and f_{16} as an example. The candidate decision tree is shown in Fig. 2. To check whether this decision tree is able to explain the decisions on 90% of the inputs throughout the input space, we apply the above-mentioned algorithm. In our experiments, we set α=β=0.05, σ=0.05 and ϕ=0.90. Then *accept_bound* and *deny_bound* are 19.0

and 0.053 respectively. After detecting 28 samples, all 28 samples have consistent prediction labels and the calculated *sprt_ratio* is 22.52 by formula 2. Since $pr \geq accpet\_bound$, we accept the hypothesis that the probability of consistency within the entire input space can reach 90%. □

4 Implementation and evaluation

Our approach has been implemented as a self-contained software toolkit based on Tensorflow (Abadi et al. 2016) and scikit-learn (Pedregosa et al. 2011). It is implemented with a total of about 4K lines of Python code. Our experiments are based on the following datasets which have been used as evaluation subjects in multiple previous studies (Agarwal et al. 2018; Zhang et al. 2020).

- **Census (Kohavi et al.**1996) The Census income dataset was published in 1996. The prediction task is to determine whether the income of an adult is above $50,000 annually based on his/her personal information. The dataset contains 32561 samples, each of which has 13 features.
- **Bank Marketing** The dataset came from a Portuguese banking institution and is used to train models for predicting whether the client would subscribe a term deposit based on his/her information. The size of the dataset is more than 45,000 and each record contains 16 attributes.
- **German Credit (Hofmann**1994) This is a small dataset with 600 samples, each of which has 20 features. The task is to assess an individual's credit based on personal and financial records.

For each dataset, we train a neural network using the exact same configuration as reported in the previous studies (Huchard et al. 2018; Zhang et al. 2020). The details are shown in Table 1. All these neural networks contain six layers. Each hidden layer contains 64, 32, 16, 8 and 4 units. The output layer contains 2 (number of predict classes) units. ReLU is used as the active function. Lastly, the Softmax function is used to output prediction probabilities. Although we focus on feedforward neural networks on classification tasks in our study, our algorithms work for more complex neural networks such as convolutional neural networks (CNNs) in general. Furthermore, we focus on neural networks trained on tabular feature vectors instead of complex data such as images and texts. This is because, in order to explain the decisions of neural networks trained on images and texts, we must additionally address

Table 1 Dataset and models of experiments

Dataset	Model	Accuracy (%)	Attributes
Census income	Six-layer Fully-connected NN	86.13	13
Bank marketing	Six-layer Fully-connected NN	91.62	16
German credit	Six-layer Fully-connected NN	100	20

the challenge of identifying high-level human-understandable features before we can learn the decision trees. Note that high-level feature extraction for such complicated data is itself an active research field (Latif et al. 2019; Deshwal et al. 2019).

In the following, we report the evaluation results which are conducted to answer multiple research questions (RQ). Note that Algorithm 4 and Algorithm 7 require multiple hyper-parameters, whose values are either determined empirically (such as the number of clusters and the max number of iterations for perturbation in Algorithm 4) or adopted from standard practice (such as α and β for hypothesis testing). The details are shown in Table 2. Here *p_num* is set to be 1000 for **Census** and **Bank** data and 600 for **Credit** data due to its small size. All experiments are conducted on a laptop running macOS (10.15.6) with 16 GB memory. Each experiment is set with a timeout of 1000 hours. All models and experiment details are available online at Zhang (2021).

RQ1: Are our algorithms efficient on testing the models' decision explainability? This question is designed to evaluate the efficiency of our algorithms, particularly the effect of the design parameters such as K and ϕ. To answer this question, we systematically apply our algorithms to the above-mentioned models and measure the results including efficiency. That is, we test each model against K-explainability for different K against the training set and the entire input space; and test each model against K, ϕ-explainability with different K and ϕ. The results of testing K-explainability are summarized in Table 3.

The third column shows the testing results and time taken for all three models with respect to different K on the training set. For all models, we can evaluate the models' K-explainability with a K value of 2 or 3. Note that in this setting, the optimal decision tree on the training set is generated based on all instances in the training set for each combination of K features, which is time-consuming. As a result, we can test 4 or 5-explainability only on the model trained based on the **Credit** dataset. In our experiments, more than 99% of the time is spent on generating the decision trees. One way to reduce time consumption is to parallelize the generation of decision trees for different combinations of features.

The fourth column shows the testing results and time taken for all inputs. We sample a threshold number of instances to build a candidate decision tree. The results here are based on training candidate models with 5000 random instances. Note that given the relatively small number of instances for building the candidate tree, we can finish testing up to 5-explainability for all models before timeout and

Table 2 Parameters of the experiments

Parameter	Value	Description
c_num	4	Cluster count
max_iter	10	Max number of iterations for perturbation
step_size	1	Step size of perturbation
p_num	1000,600	Number of seed instances for perturbation
α, β	0.05	Error probability bounds
σ	0.05	Indifference region

Table 3 *K*-explainability testing results

Dataset	K	Testing training set		Testing all inputs	
		Result	Time (min)	Result	Time (min)
Census	2	No	370.01	No	33.68
	3	No	12485.24	No	223.27
	4	No	T/O	No	685.57
	5	No	T/O	No	2110.25
Bank	2	No	5538.10	No	45.60
	3	No	51647.40	No	384.63
	4	No	T/O	No	1980.77
	5	No	T/O	No	6149.42
Credit	2	No	0.48	No	241.77
	3	No	7.22	No	1470.45
	4	No	33.12	No	6456.20
	5	No	121.17	No	28250.87

the number of random instances may have an effect on the quality of the candidate models. The number 5000 is determined empirically based on the experiment results shown in Table 4. It shows the time taken of generating one decision tree as well as the maximum accuracy ϕ achieved by the candidate model on the **Census** dataset. It can be observed that the accuracy achieved by the decision tree often maximizes when the number of instances is 5000. Further increasing the number of instances increases the time proportionally without increasing the accuracy. In multiple cases, the accuracy even drops. Thus, we set the 5000 as a threshold to learn candidate decision trees in testing against all inputs.

The results of testing K, ϕ-explainability against different K and ϕ as well as the training set or all inputs are summarized in Table 5. ϕ is set to be 5 values, i.e.,

Table 4 Training time with different #instances

K	#instances	T_{dt}(s)	$max(\phi)$
2	2000	4.20	93%
	5000	25.74	93%
	10000	112.11	93%
3	2000	9.06	95%
	5000	46.66	96%
	10000	133.21	95%
4	2000	15.17	95%
	5000	57.34	97%
	10000	140.94	95%
5	2000	21.32	96%
	5000	98.15	97%
	10000	202.06	97%

Table 5 K, ϕ-explainability testing result

Dataset	K	Testing against training set										Testing against all inputs									
		70%		80%		90%		95%		99%		70%		80%		90%		95%		99%	
		Re.	Time	Re.	Time	Re.	Time	Re.	Time	Re.	Time	Re.	Time	Re.	Time	Re.	Time	Re.	Time	Re.	Time
Census	2	Yes	4.75	Yes	4.75	No	370.02	No	370.02	No	370.02	Yes	0.43	Yes	6.75	Yes	33.78	No	33.83	No	33.79
	3	Yes	43.69	Yes	43.69	No	12485.33	No	12485.33	No	12485.33	Yes	1.75	Yes	44.70	Yes	223.42	Yes	223.42	No	223.38
	4	Yes	T/O	Yes	T/O	Yes	T/O	No	T/O	No	T/O	Yes	0.96	Yes	137.14	Yes	411.43	Yes	480.02	No	685.69
	5	Yes	T/O	Yes	T/O	Yes	T/O	No	T/O	No	T/O	Yes	1.64	Yes	422.08	Yes	1266.25	Yes	1582.82	No	2110.37
Bank	2	Yes	46.15	Yes	46.15	Yes	46.15	No	5538.14	No	5538.14	Yes	0.38	Yes	0.38	Yes	38.89	No	45.78	No	45.74
	3	Yes	92.23	Yes	92.23	Yes	92.23	Yes	92.23	No	17523.23	Yes	0.69	Yes	1.67	Yes	192.39	Yes	373.27	No	384.78
	4	Yes	T/O	Yes	T/O	Yes	T/O	Yes	T/O	No	T/O	Yes	1.09	Yes	1.09	Yes	990.47	Yes	1723.41	No	1980.91
	5	Yes	T/O	Yes	T/O	Yes	T/O	Yes	T/O	No	T/O	Yes	1.41	Yes	1.41	Yes	3647.79	Yes	5780.61	No	6149.59
Credit	2	Yes	0.48	No	0.48	No	0.48	No	0.48	No	0.48	Yes	26.69	No	241.79	No	241.79	No	241.79	No	241.79
	3	Yes	0.63	No	7.22	No	7.22	No	7.22	No	7.22	Yes	242.50	No	1470.48	No	1470.48	No	1470.48	No	1470.48
	4	Yes	0.68	No	33.17	No	33.17	No	33.17	No	33.17	Yes	1700.34	No	6456.23	No	6456.23	No	6456.23	No	6456.22
	5	Yes	0.78	No	121.30	No	121.30	No	121.30	No	121.30	Yes	8394.73	No	28250.90	No	28250.90	No	28250.90	No	28250.89

70%, 80%, 90%, 95%, and 99%. In the third column, we show the testing results and time taken for evaluating the fidelity of all decision trees against the training set. Note that we can still measure prediction accuracy on decision trees generated before timeout. The fourth column shows the results and time taken on testing K, ϕ -explainability against all inputs.

We note that the execution time is dominated by the time required for learning the decision trees. Since we consider all feature combinations with different size K, the number of all possible feature sets would be large especially with regards to some training set with high-dimensional feature vectors. In the case of testing K, ϕ -explainability against all inputs, it is strongly related to the number of feature combinations and the number of samples we use to train the candidate decision trees.

One practical way to reduce the complexity is thus to heuristically select a subset of the features, i.e., those which are likely correlated to the neural network's decision. For example, SHapley Additive explanation uses Shapely values to compute the contributions of the features (Lundberg et al. 2017). Based on the Shapely values, we can focus on the features with high contribution to generate the decision trees. In the following, we conduct experiments to evaluate this idea by focusing on the top 6 contributing features based on Shapely value ranking. That is, we only consider the decision trees with the selected features and test each model against k, ϕ -explainability against the training set and the entire input space. The reason why we select 6 features is that the maximum K in our experiments is 5. The testing results are summarized in Table 6.

We can observe that, although we focus on the top 6 features only, the testing results (i.e., whether each model satisfies the corresponding explainability metric) remain exactly the same as the results in Table 5. The time cost however decreases significantly, especially for the first two neural networks trained on

Table 6 K, ϕ-explainability testing result based on Shapley values

Dataset	K	Testing against training set									
		70%		80%		90%		95%		99%	
		Re.	Time	Re.	Time	Re.	Time	Re.	Time	Re.	Time
Census	2	Yes	5.04	Yes	5.04	No	37.24	No	37.24	No	37.24
	3	Yes	5.34	Yes	5.34	No	41.86	No	41.86	No	41.86
	4	Yes	6.87	Yes	6.87	Yes	33.55	No	70.87	No	70.87
	5	Yes	5.11	Yes	5.11	Yes	5.11	No	29.25	No	29.25
Bank	2	Yes	4.16	Yes	4.16	Yes	41.82	No	41.82	No	41.82
	3	Yes	5.31	Yes	5.31	Yes	46.05	Yes	46.05	No	46.05
	4	Yes	5.94	Yes	5.94	Yes	29.63	Yes	44.31	No	44.31
	5	Yes	6.24	Yes	6.24	Yes	6.24	Yes	27.11	No	27.11
Credit	2	Yes	1.53	No	19.06	No	19.06	No	19.06	No	19.06
	3	Yes	5.45	No	15.77	No	15.77	No	15.77	No	15.77
	4	Yes	2.96	No	20.28	No	20.28	No	20.28	No	20.28
	5	Yes	4.03	No	11.22	No	11.22	No	11.22	No	11.22

Census dataset and **Bank** dataset. When K is set as 4 or 5, the time cost reduces from T/O to no more than 71 minutes. Furthermore, the execution time for most tests is far less than 1 hour.

RQ2: Are existing models' decisions explainable? This question aims to apply our approach to study decision explainability of neural network models. To answer the question, we investigate the decision explainability of the models using our approach based on the results shown in Table 3 and Table 5.

In terms of K-explainability, it can be observed that every model is found to be un-explainable, whether considering only those in the training set or all inputs. In other words, no matter what combination of K-features used to explain the decisions of the neural network, there are always counterexamples, i.e., instances which are predicted differently by the decision tree and the neural network. In particular, the adversarial sampling approach adopted in Algorithm 4 is proved to be effective in identifying such counterexamples. These results show that indeed it is unlikely that we can always explain the decisions of a neural network using a decision tree. Thus, the goal should perhaps be minimizing the percentage of such un-explainable cases.

In terms of K, ϕ-explainability test, the results shown in Table 5 are mixed. We have several observations. First, comparing the results with different K values, the bigger the K is, the bigger a ϕ can be achieved. This is intuitively reasonable since it is easier to explain the prediction of a neural network with a more complicated decision tree. Second, it is not necessarily easier to explain the instances in the training set than to explain all input instances. Note that there are instances where a model fails a K, ϕ-explainability test on the training set but passes the test on the entire input space. For instance, the models trained on **Census** dataset pass the test with higher ϕ against all inputs than against the training set. In a close investigation, we discover that this training set is highly imbalanced, e.g., samples with one label are significantly more than samples within the other labels. Neural networks trained on such an imbalanced dataset are known to produce imbalanced predictions, e.g., the majority of predictions on random samples are the same label. The majority of the randomly generated samples of Census dataset are predicted as label 1. Such imbalanced predictions are much easier to explain, i.e., it can be explained with a simple decision tree which always generates the same prediction (e.g., "everyone's application is rejected"). This is confirmed in our experiments, most of the randomly generated samples are predicted with the same label. As a result, even with a K value of 3, a ϕ of 95% can be achieved. Furthermore, in our experiments, we test K, ϕ-explainability against all inputs using the SPRT algorithms, since it is infeasible to enumerate all inputs. The SPRT algorithm provides only statistically results with a bounded range of errors, which is more "relaxed" compared to testing against all samples in the training set.

Lastly, it can be seen that the results vary across different models. For instance, the neural network trained on the **Credit** dataset has the lowest decision explainability. The fidelity of the learned decision trees with respect to the training set and all inputs is less than 80% no matter what K value is used. The highest ϕ achieved by the neural network against all inputs is only 70%. Our interpretation of the result is as follows. Because this dataset is very small, the model is less robust compared

with the models trained on the other dataset. In other words, its predictions on unseen instances are rather random and thus hard to explain.

RQ3: Are robust models' decision more explainable? One of the observations in the above experiments is that Algorithm 4 is often successful in finding instances which are un-explainable by the decision tree with adversarial sampling. The above experiment results seem to suggest that the lack of robustness often makes a model's decision un-explainable. This question is thus designed to test this hypothesis, i.e., with the help of robust training, we aim to see whether more robust models' decisions are more explainable. That is, whether it becomes harder to find 'counterexamples' that have different predictions of the neural network and the decision tree after retraining. Here, we use the technique called FGSM (Goodfellow et al. 2014) to compute adversarial perturbations and retrain the model. Note that the label of the samples generated through adversarial perturbation is the same as the original sample.

To answer this question, we test the decision explainability of the retrained models using our approach and check whether a higher ϕ can be achieved. After robust training, the level of ϕ on the training set remains almost identical to that without robust training. We thus focus on K, ϕ-explainability testing against all inputs. The results are shown in Table 7, where 7 values of ϕ (i.e., 70%, 75%, 80%, 85%, 90%, 95% and 99%) are tested. We highlight improved results in green and worsened results in red.

Compared to the corresponding entries in Table 5, we observe that among the 12 cases (4 different K values on three models), 4 results show improvement and 5 results show worse decision explainability after robust training. The two models trained on the **Census** and **Bank** dataset become less explainable. For instance, the model trained on **Census** without robust training is $2, 90\%$-explainable against all inputs and it is only $2, 85\%$-explainable after training. Our hypothesis is that the two models trained on **Census** and **Bank** dataset are able to achieve high K, ϕ-explainability because the

Table 7 Results after robust training

Dataset	K	ϕ in testing against all inputs						
		70%	75%	80%	85%	90%	95%	99%
Census	2	Yes	Yes	Yes	Yes	No	No	No
	3	Yes	Yes	Yes	Yes	Yes	No	No
	4	Yes	Yes	Yes	Yes	Yes	No	No
	5	Yes	Yes	Yes	Yes	Yes	No	No
Bank	2	Yes	Yes	Yes	Yes	Yes	No	No
	3	Yes	Yes	Yes	Yes	Yes	No	No
	4	Yes	Yes	Yes	Yes	Yes	Yes	No
	5	Yes	Yes	Yes	Yes	Yes	Yes	No
Credit	2	Yes	Yes	Yes	Yes	No	No	No
	3	Yes	Yes	Yes	Yes	No	No	No
	4	Yes	Yes	Yes	Yes	No	No	No
	5	Yes	Yes	Yes	Yes	No	No	No

model makes simplified predictions as the result of imbalanced training data. After robust training, the training data (i.e., the original data plus those generated through adversarial perturbation) become relatively more balanced. As a result, the neural network model makes more complicated predictions, and thus its K, ϕ-explainability decreases. On the contrarily, the model trained on the **Credit** dataset becomes much more explainable after robust training. This can be explained by the fact that the **Credit** dataset, although small, is more balanced, and in such a case, robust training actually improves decision explainability. We acknowledge that this hypothesis needs to be evaluated with a large number of models to be conclusive.

RQ4: Can we improve model decision explainability using our testing results? Many practical scenarios would prefer models whose decisions can be explained. This *RQ* thus aims to see whether our approach could help improve model decision explainability. The idea is to check whether a neural network that fails to reach a certain level of K, ϕ-explainability can be improved through retraining with un-explainable adversarial samples identified using our approach. That is, we label those adversarial samples generated by Algorithm 4 with the labels of corresponding seed instances (i.e., the labels of the samples in training set predicted by the neural network). Then we retrain the neural network with these additional samples and apply our approach to test the decision explainability of the retrained models. Note that the premise condition of Algorithm 4 is that all samples x in the original labeled dataset LD satisfy $D(x) = N(x)$ (as shown in line 6 at Algorithm 3). That is, all seed instances x in Algorithm 4 have the same predictions by the neural network and decision tree. Here, we assume the generated adversarial sample x' has the same ground truth label with the original seed instance x. The testing results as well as the accuracy of retrained models are shown in Table 8. We highlight improved results in green as well. Note that 6 results show improvement after retraining and none shows worsened decision explainability.

We observe that all three models' decisions become more explainable after retraining. For instance, the models trained on **Census** and **Bank** dataset are both

Table 8 Results after training with 'counterexamples'

Dataset	Accuracy	K	ϕ in testing against all inputs						
			70%	75%	80%	85%	90%	95%	99%
Census	84.91%	2	Yes	Yes	Yes	Yes	Yes	Yes	No
		3	Yes	Yes	Yes	Yes	Yes	Yes	No
		4	Yes	Yes	Yes	Yes	Yes	Yes	No
		5	Yes	Yes	Yes	Yes	Yes	Yes	No
Bank	91.53%	2	Yes	Yes	Yes	Yes	Yes	Yes	No
		3	Yes	Yes	Yes	Yes	Yes	Yes	No
		4	Yes	Yes	Yes	Yes	Yes	Yes	No
		5	Yes	Yes	Yes	Yes	Yes	Yes	No
Credit	90.3%	2	Yes	Yes	Yes	Yes	No	No	No
		3	Yes	Yes	Yes	Yes	No	No	No
		4	Yes	Yes	Yes	Yes	No	No	No
		5	Yes	Yes	Yes	Yes	No	No	No

2, 90%-explainable against all inputs before retraining. After retraining, both of the two models achieve 2, 95%-explainable. The model trained on **Credit** can achieve ϕ as 85% for any given K after retraining. The results thus proof that our testing algorithms could be potentially used to improve model decision explainability, by paying a relatively small price in terms of accuracy. Although the improvement in the model trained on **Credit** is more substantial, the accuracy drops mostly from 100% to 90.3% compared with the models trained on **Census** and **Bank** dataset. This result suggests that higher explainability may come at the cost of prediction accuracy. More precise trade-offs between the neural network's explainability and accuracy need further experiments.

Threats to validity In the experiment, only 3 datasets are applied to evaluate the effectiveness of our approach. This could be further improved with more models and datasets as well as protected features. Furthermore, in the experiments, we focus on feed-forward neural networks only. Our approach can be potentially adopted for other neural network models. Lastly, the datasets are all tabular data. For complicated models such as RNN for text, the task is likely more complicated.

5 Related work

The term explainability or interpretability has been used to refer to multiple different things. For instance, Murdoch *et al.* attempted to define interpretability in the context of machine learning and placed it as a part of the generic data science life cycle (Murdoch et al. 2019). They defined interpretable machine learning as the use of machine learning models for the extraction of relevant knowledge in data. Montavon *et al.* defined interpretation as the mapping of an abstract concept into a domain that the human can make sense of. Examples of interpretable domains are images or texts (Montavon et al. 2018). Models from domains with abstract vector spaces are deemed to be un-interpretable. The notion of explainability or interpretability in the above studies is informal and not measurable, i.e., there is no way to quantitatively measure or compare models in terms of explainability or interpretability.

The term interpretability is often associated with studies which aim to provide hints on how neural networks work internally, such as studies on local/global interpretability of neural networks. Studies on local interpretability focus on investigating neural network prediction on one or a specific set of samples. One of the popular methods is the saliency map approach (Zeiler et al. 2014). The idea is to identify specific parts of an input sample that contribute the most to the prediction of the network (Sundararajan et al. 2017; Dabkowski and Gal 2017) or the activity of a specific layer in the network (Zhang et al. 2019). In the case of image classifier interpretation, it is useful to know which parts of the input activate certain filters for the prediction and how each part contributes to the prediction score at the pixel level (Lundberg et al. 2017) or at the object level (Zhang et al. 2019). In Kim et al. (2018), proposed TCAV which quantitatively tests the contribution of user-defined concepts. The difference between these approaches and ours is that these approaches focus on explaining prediction on one particular input, whereas we aim to provide a way of measuring neural networks' decision explainability as a whole.

Studies on the global interpretability of neural networks aim to come up with models that are simple enough to be human-understandable and yet expressive enough to predict the predictions of the neural network (at least in most of the cases). Such candidates include regression models (Schielzeth 2010), decision trees (Loh 2011), decision rules (Lakkaraju et al. 2016). In Lakkaraju et al. (2016), proposed interpretable decision sets, i.e., a framework for building high-accurate predictive models, yet also interpretable. They capture the interpretability of a decision set by defining four natural metrics: size, length, cover and overlap. They also set metrics to measure the accuracy of each rule. While these approaches share the same idea of using simple models to mimic neural networks, they do not provide ways of testing and measuring the degree of decision explainability.

This work is also related to work on testing neural networks. Unlike traditional software systems that have clear and controllable logic, the lack of interpretability of neural networks makes system testing difficult. In Pei et al. (2017), introduced the concept of neuron coverage for measuring testing coverage of a neural network. They considered a neuron to be activated if its output is higher than a threshold value and unactivated otherwise. They generated neurons' activation status and a set of test inputs based on the number of neurons activated by the inputs and propose a number of alliterative coverage metrics. In terms of fairness testing, Kusner *et al.* introduced a causal approach to address fairness. They leveraged the causal framework to model the relationship between protected attributes (Kusner et al. 2017). This work can work as an assisting tool to fairness testing. It can analyze whether the model's decisions are relevant to sensitive features. If the explainable decision tree with high fidelity contains propositions with certain sensitive attributes, the model is likely unfair. Once this work helps understanding predictions and behaviors of neural networks, it might help with further improvement as well. To the best of our knowledge, this is the first approach on testing neural networks' decision explainability.

6 Conclusion

In this work, we propose multiple definitions of neural network interpretability and develop algorithms to systematically test the decision explainability of neural networks. We define decision explainability based on measuring its fidelity against decision trees with a height limit. We remark that this is an initial attempt at testing model decision explainability and there is much to be done in the future.

Acknowledgements This research/project is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2019-012).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: Tensorflow: A system for large-scale machine learning. In: 12thfUSENIXg symposium on operating systems design and implementation (fOSDIg 16). pp. 265–283(2016)

Agarwal, A., Lohia, P., Nagar, S., Dey, K., Saha, D.: Automated test generation to detect individual discrimination in AI models'. In: arXiv preprint arXiv:1809.03260 (2018)
Bojarski, M., Testa, DD., Dworakowski, D.,Firner, B., Flepp, B., Goyal, P., Jackel, LD., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars'. In: arXiv preprint arXiv:1604.07316 (2016)
Breiman, L.: Jerome Friedman, Charles J Stone, and Richard A Olshen. Classification and regression trees. CRC press, (1984)
Carvalho, D.V., Pereira, E.M., Cardoso, J.S.: Machine learning interpretability: a survey on methods and metrics. In: Electron. **8**(8), 832 (2019)
Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system'. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. , pp. 785–794 (2016)
Clauset, A.: A brief primer on probability distributions'. In: Santa Fe Institute. (2011)
Dabkowski, P., Gal, Y.: Real time image saliency for black box classifiers. In: Advances in Neural Information Processing Systems. , pp. 6967–6976(2017)
David, W.J.A.D.: A single-parameter generalization of the Gini indices of inequality. In: J. Econom. Theory **22**(1), 67–86 (1980)
Deshwal, D., Sangwan, P., Kumar, D.: Feature extraction methods in language identification: a survey. In: Wireless Personal Commun. **107**(4), 2071–2103 (2019)
Goodfellow, IJ., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples'. In: arXiv preprint arXiv:1412.6572 (2014)
Guo, P., Anderson, C., Pearson, K., Farrell, R.: Neural network interpretation via fine grained textual summarization. In: arXiv preprint arXiv:1805.08969 (2018)
Hofmann, H.: German credit dataset. In: (1994). https://archive. ics . uci . edu / ml / datasets / statlog + (german + credit + data)
Huchard, M., Kästner, C., Fraser, G.: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018). In: ASE: Automated Software Engineering. ACM Press. (2018)
Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., Liu, T.: Lightgbm: A highly efficient gradient boosting decision tree'. In: Advances in neural information processing systems. pp. 3146–3154 (2017)
Kim, Been., Rudin, Cynthia., Shah, Julie A.:"The bayesian case model: A generative approach for case-based reasoning and prototype classification". In: Advances in neural information processing systems. pp. 1952–1960(2014)
Kim, B., Wattenberg, M., Gilmer, J., Cai, C., Wexler, J., Viegas, F., et al.:Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In: International conference on machine learning. pp. 2668–2677 (2018)
Kohavi, BBR.:Data Mining and Visualization. In: (1996). https://archive.ics.uci.edu/ml/datasets/adult
Kurakin, A., Goodfellow, I., Bengio, S.: dversarial examples in the physical world. In: arXiv preprint arXiv:1607.02533 (2016)
Kusner, MJ., Loftus, J., Russell, C., Silva, R.: Counterfactual fairness". In: Advances in neural information processing systems. pp. 4066–4076(2017)
Lage, I., Chen, E., He, J., Narayanan, M., Kim, B., Gershman, S., Doshi-Velez, F.: An evaluation of the human-interpretability of explanation. In: arXiv preprint arXiv:1902.00006 (2019)
Lakkaraju, H., Bach, SH., Leskovec, J.: nterpretable decision sets: A joint framework for description and prediction. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. pp. 1675–1684(2016)
Latif, Afshan., R, Aqsa., S, Umer., A, Jameel., A, Nouman., R, Naeem I., Zafar, B., Dar, SH., Sajid, M., Khalil, T.:Content-based image retrieval and feature extraction: a comprehensive review. In: Mathematical Problems in Engineering 2019 (2019)
Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.:Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In: arXiv preprint arXiv:1910.13461 (2019)
Liao, C., Zhong, H., Squicciarini, A., Zhu, S., Miller, D.:Backdoor embedding in convolutional neural network models via invisible perturbation. In: arXiv preprint arXiv:1808.10307 (2018)
Lloyd, S.: Least squares quantization in PCM. In: IEEE Trans. Inform. theory **28**(2), 129–137 (1982)
Loh, W.-Y.: Classification and regression trees. In: Wiley Interdis. Rev.: Data Mining and Knowledge Dis. **1**(1), 14–23 (2011)

Lombrozo, T.: The structure and function of explanations. In: Trends in Cognitive Sci. **10**(10), 464–470 (2006)
Lombrozo, T.: Simplicity and probability in causal explanation. In: Cognitive Psychol. **55**(3), 232–257 (2007)
Lundberg, SM., Lee, S.: unified approach to interpreting model predictions. In: Proceedings of the 31st international conference on neural information processing systems. pp. 4768–4777(2017)
Mohammed, M.A., Ghani, M.K.A., Arunkumar, N., Hamed, R.I., Mostafa, S.A., Abdullah, M.K., Burhanuddin, M.A.: Decision support system for nasopharyngeal carcinoma discrimination from endoscopic images using artificial neural network. In: J Supercomput. **76**(2), 1086–1104 (2020)
Montavon, G., Samek, W., Müller, K.-R.: Methods for interpreting and understanding deep neural networks. In: Digit. Sig. Process **73**, 1–15 (2018)
Moosavi-Dezfooli, SM., Fawzi, Al., Frossard, P.:Deepfool: a simple and accurate method to fool deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2574–2582(2016)
Moro, S., Cortez, P., Rita, P.:A data-driven approach to predict the success of bank telemarketing. In: Decision Support Systems 62 (2014). https://archive.ics.uci.edu/ml/datasets/bank+marketing, pp. 22–31
Murdoch, W J., Singh, C., Kumbier, K., Abbasi-Asl, R., Yu, B.:Interpretable machine learning: definitions, methods, and applications. In: arXiv preprint arXiv:1901.04592 (2019)
Oaksford, M., Chater, N.: The probabilistic approach to human reasoning. In: Trends in Cognit. Sci. **5**(8), 349–357 (2001)
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: machine learning in Python. In: J. Mach. Learn. Res. **12**, 2825–2830 (2011)
Pei, K., Cao, Y., Yang, J., Jana, S.:Deepxplore: Automated whitebox testing of deep learning systems. In: proceedings of the 26th Symposium on Operating Systems Principles. , pp. 1–18(2017)
Schielzeth, H.: Simple means to improve the interpretability of regression coefficients. In: Methods in Ecol Evol **1**(2), 103–113 (2010)
Shore, H.: Approximating an unknown distribution when distribution information is extremely limited. In: Commun. Statistics-Simulation and Comput. **27**(2), 501–523 (1998)
Simonyan, K., Vedaldi, A., Zisserman, A.:Deep inside convolutional networks: Visualising image classification models and saliency maps. In: arXiv preprint arXiv:1312.6034 (2013)
Steinberg, D., Colla, P.: CART: classification and regression trees. In: The top ten Algorithms in Data Min **9**, 179 (2009)
Sundararajan, M., Taly, A., Yan, Q.:Axiomatic attribution for deep networks. In: arXiv preprint arXiv:1703.01365 (2017)
Tran, L., Yin, X., Liu, X.:Disentangled representation learning gan for pose-invariant face recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1415–1424(2017)
Wald, A.: Sequential tests of statistical hypotheses. In: Ann. Math. Statistics **16**(2), 117–186 (1945)
Wald, A., Wolfowitz, J.: Optimum character of the sequential probability ratio test. In: The Annals of Mathematical Statistics pp. 326–339(1948)
Yosinski, Jason., Clune, Jeff., Nguyen, Anh., Fuchs, Thomas., Lipson. Hod .:"Understanding neural networks through deep visualization". In: arXiv preprint arXiv:1506.06579 (2015)
Zeiler, Matthew D., Fergus, R.: Visualizing and understanding convolutional networks. In: European conference on computer vision. Springer. pp. 818–833(2014)
Zhang, M.:GitHub Repository. In: (2021). https://github.com/zhangmengling/NN_interpretability.git
Zhang, P., Wang, J., Sun, J., Dong, G., Wang, X., Wang, X., Dong, JS., Dai, T.: White-box Fairness Testing through Adversarial Sampling. In: Proceedings of the 42rd International Conference on Software Engineering (ICSE 2020), Seoul, South Korea (2020)
Zhang, P., Hou, Y., Song, D.: Approximating true relevance distribution from a mixture model based on irrelevance data. In: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. pp. 107–114(2009)
Zhang, Q., Yang, Y., Ma, H., Wu, YN.: Interpreting cnns via decision trees. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6261–6270 (2019)